EDN ACCESS
THE DESIGN INFORMATION SOURCE OF THE ELECTRONICS INDUSTRY

Design Ideas: *May 9, 1996*

Program provides ISA-bus DLL for Windows

Paul Kemp
NASA Johnson Space Center, Houston, TX

Windows commonly uses a dynamic link library (DLL) to create subroutines and procedures that all compatible software can call. A DLL is an executable file, with extension .DLL, that a program can load and unload as required. However, not all Windows-based software can read from and write to the ISA bus in a PC. The DLL described here provides 8- and 16-bit I/O reads and writes to the ISA bus. The reason for developing the DLL is because Microsoft's Visual Basic currently has no I/O-bus read-and-write capability.

The DLL, ISACOMM.DLL, is written with Borland's C++, because this language has the ability to compile subroutines and functions as DLLs for Windows. C++ allows you to embed assembly-language instructions in the program. This ability is beneficial, because one goal of a DLL is to execute instructions as quickly as possible. The write routines of the DLL simply use the "out" command associated with the x86 family of µPs. The read routines use the "in" command. The subroutine for writing a byte to the ISA bus is as follows:

```
pusha           // Save all registers to stack so no previous
                register values are lost
mov dx,busaddr  // Load bus address (provided by user's soft
                ware calling this DLL) into DX
mov ax,busvalue //Load data byte value (also passed to DLL)
                into AX
out dx,al       // Perform I/O bus write command (use AL
                not AX because byte transfer)
popa            // Restore all register values before this DLL
                was called to registers
```

You can call this 8-bit I/O bus write subroutine from Visual Basic by placing the following line in the general-declarations portion of a form:

Declare Sub ISAWrite Lib "put DOS path of location of DLL here"(ByVal BusAddr As Integer, ByVal busvalue As Integer)

In a declare statement, everything must be on one line. Once the DLL has been declared, Visual Basic's Call command can call it. An example is the line of code used to write FFH to

I/O-bus address 0300H:

Call ISAWrite8(&H300, &HFF)

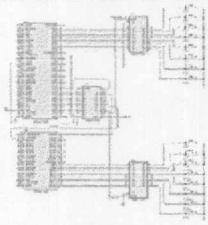
The DLL also contains the following functions and subroutines for reading and writing the I/O bus:

ISAWrite16(busaddress, busvalue) // Writes a word to the I/O bus
ISARead8(busaddress) // Reads a byte from the I/O bus
ISARead16(busaddress) // Reads a byte from the I/O bus

You can read a byte from the ISA bus by declaring the function "ISARead8" and using a variable declared as type "Long" (long integer) to store the 8-bit bus value. The following example reads I/O address 0304H and stores the 8-bit bus value in a variable named "BusData":

BusData=ISARead8(&H304)

You'll find an executable Visual Basic program, "ISACOMM.EXE," the source code, and other utilities in the compressed ZIP file attached to DI_SIG#1760 on EDN's home page at www.ednmag.com. The program allows you to read and write the I/O bus in both 8- and 16-bit modes. To run this program, you must place the files VBRUN300.DLL and ISACOMM.EXE in the Windows System directory (C:\WINDOWS\SYSTEM). Figure 1 shows the Windows-based user interface. Figure 2 gives the OrCAD-SDT-generated schematics for the AT-bus breadboards used for testing the DLL. Data I/O's Abel PLD produced the JEDEC files for the bus-address decoders used in the AT-bus breadboards. (DI #1760)



| EDN Access | feedback | **subscribe to EDN!** |
| design features | out in front | design ideas | departments | products | columnist |